1/33

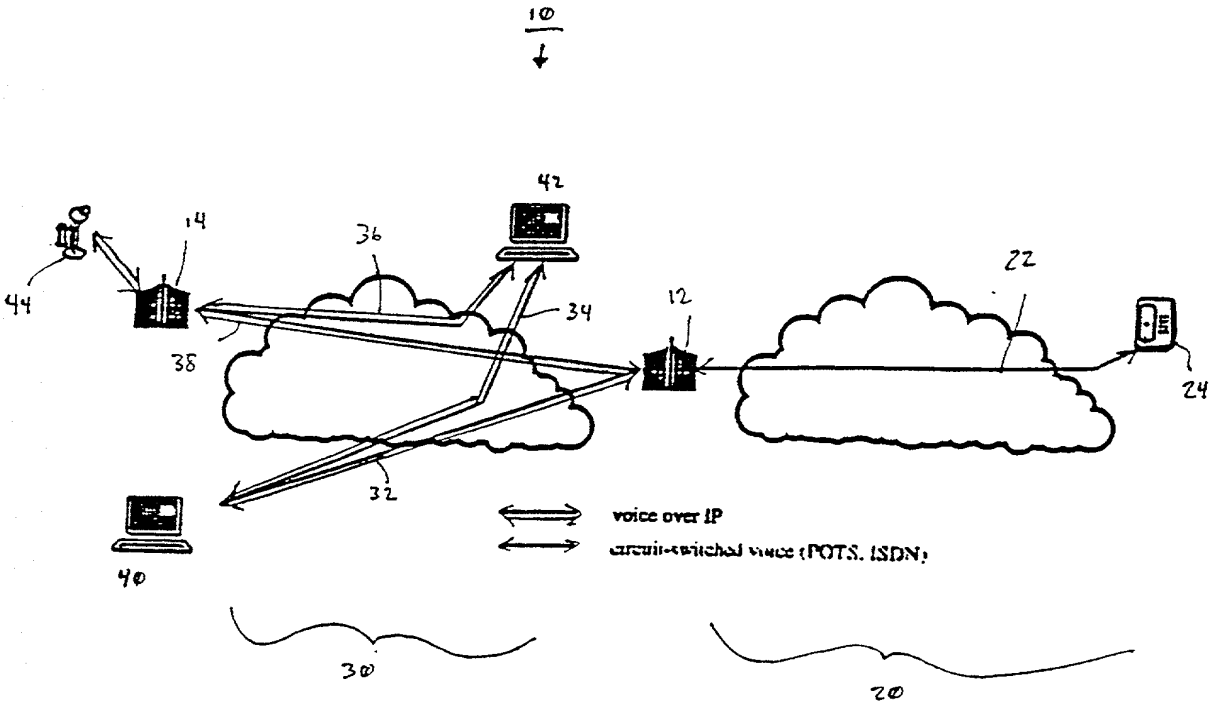


FIG. 1

2/33

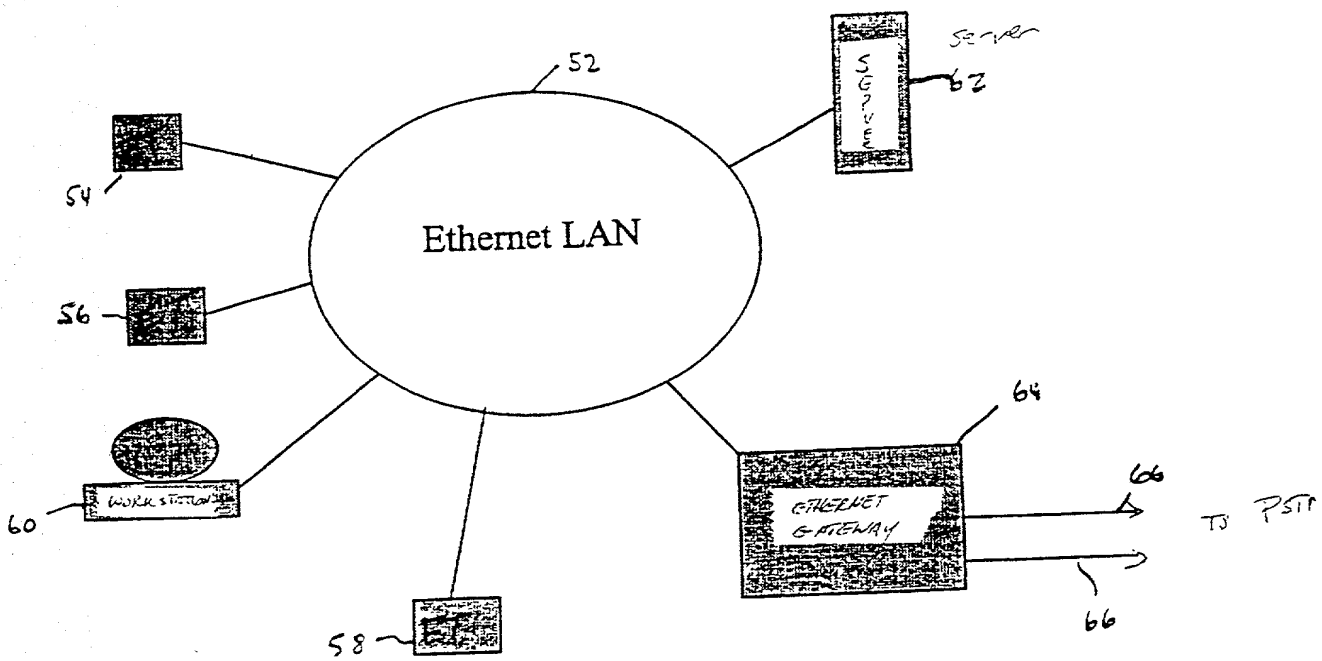

52
Ethernet LAN
SERVER
62
54
56
60
WORKSTATION
58
64
ETHERNET GATEWAY
66
66
TO PSTN

FIG 2

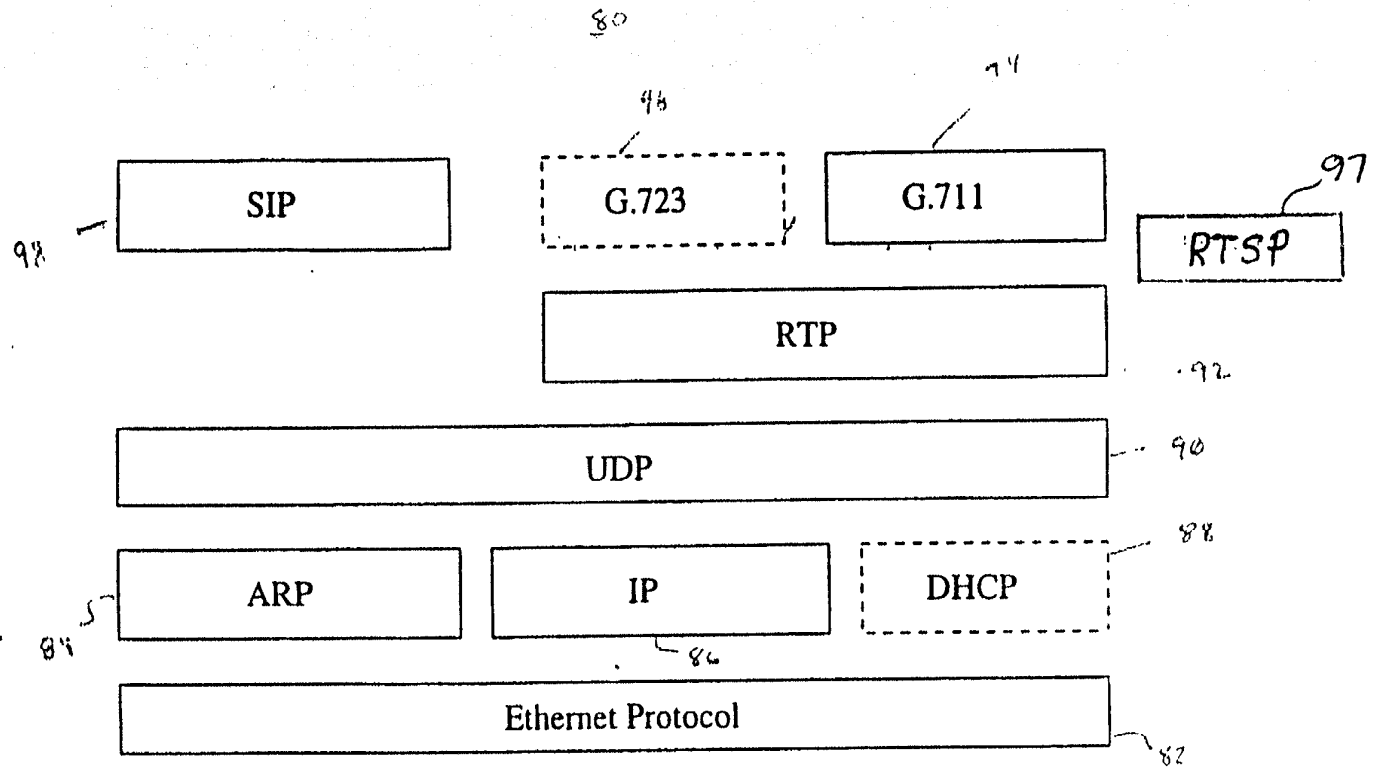


FIG 3

100

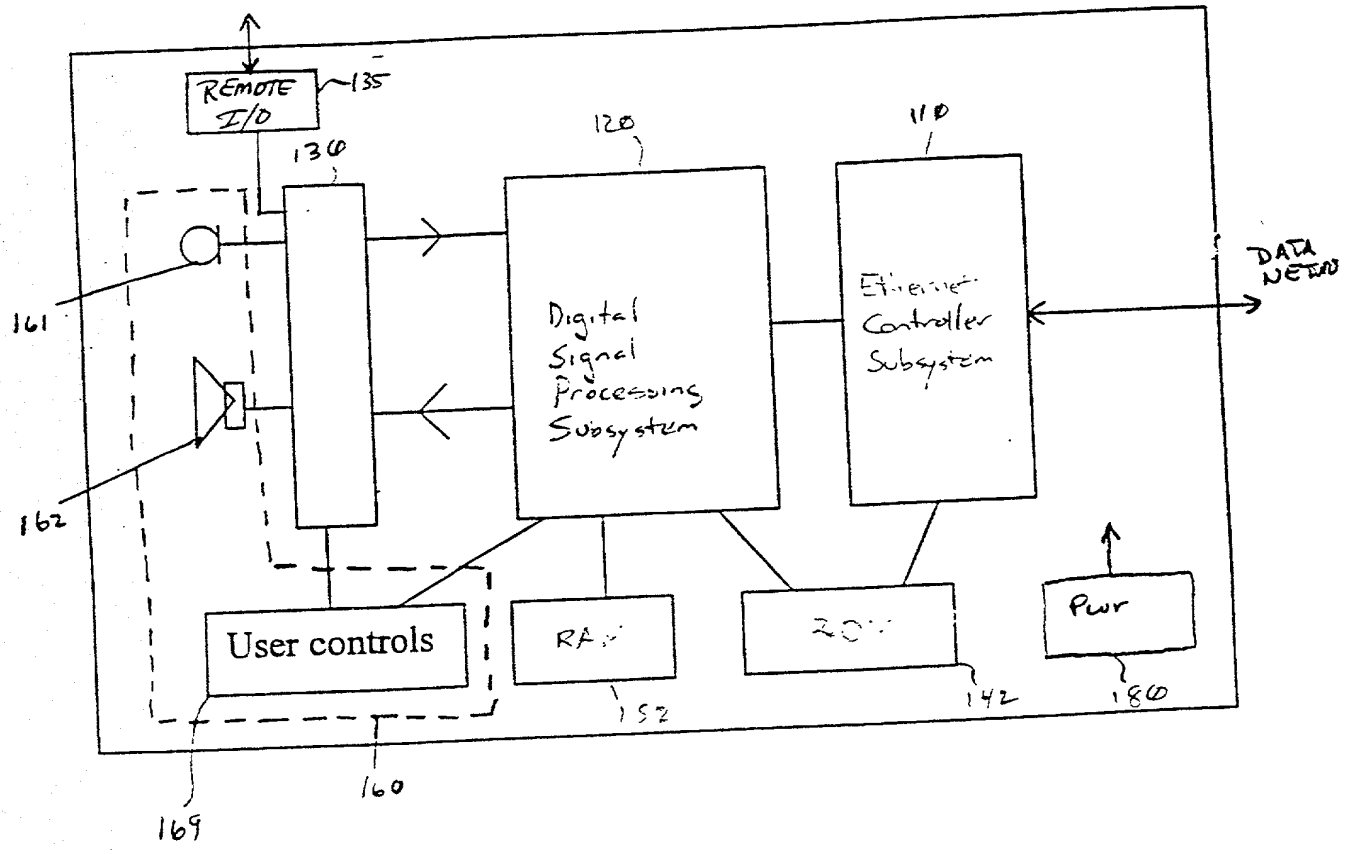

REMOTE I/O
135
130
120
110
Digital Signal Processing Subsystem
Ethernet Controller Subsystem
DATA NETWORK
161
162
User controls
RAM
152
142
Pwr
180
160
169

FIG. 4

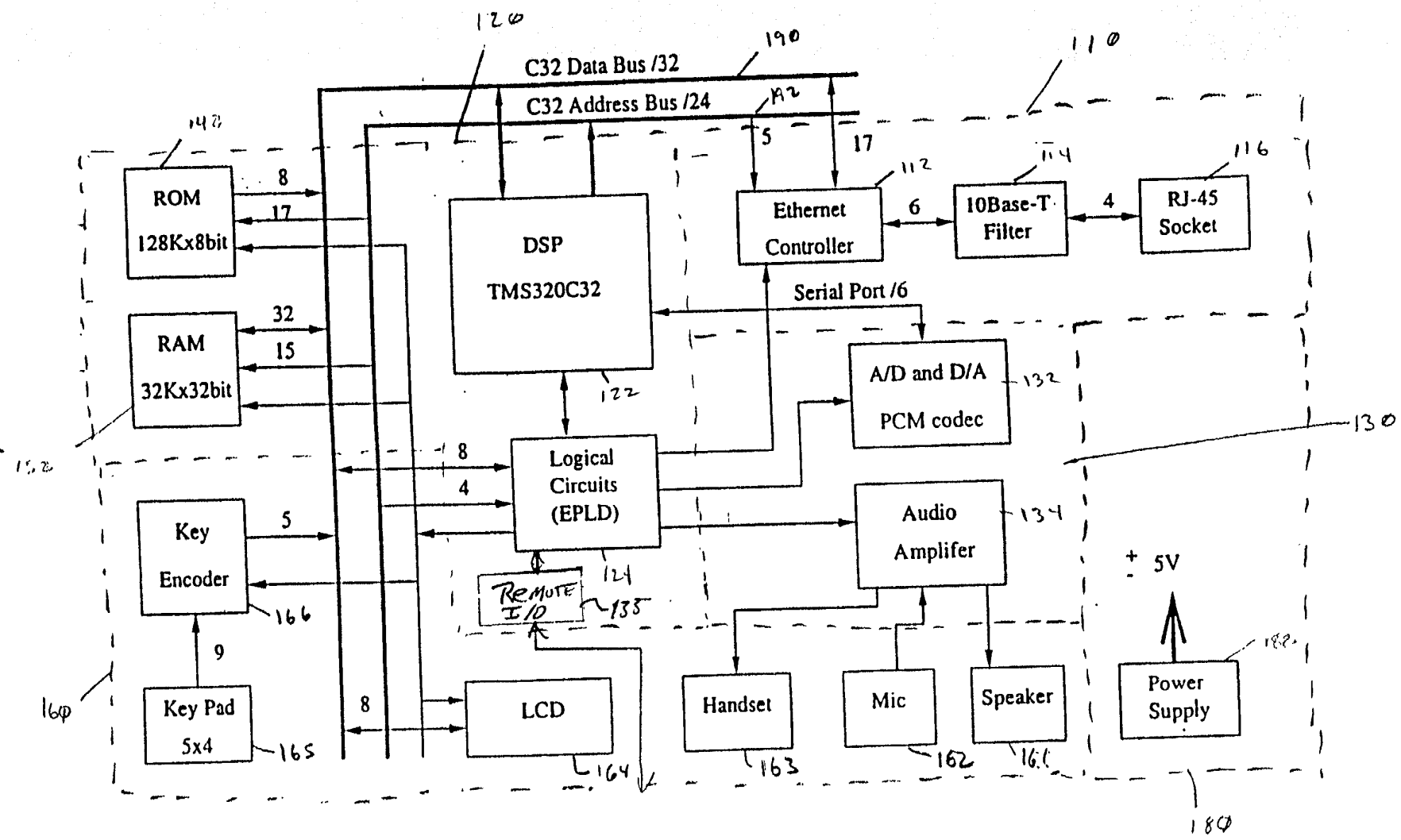


FIG. 5

DSP Address	Length (hex)	(dec.) Usage
0x1000	0xB000	128K ROM
0x20000	0x10	16 LCD
0x810000	0x20	32 Ethernet controller
0x820000	0xf	16 Keypad read, audio amplifier, control, hook control, system software reset
0x87fe00	0x200	512 Internal RAM
0x900000	0x8000	32K External Ram

FIG. 6 DSP memory map

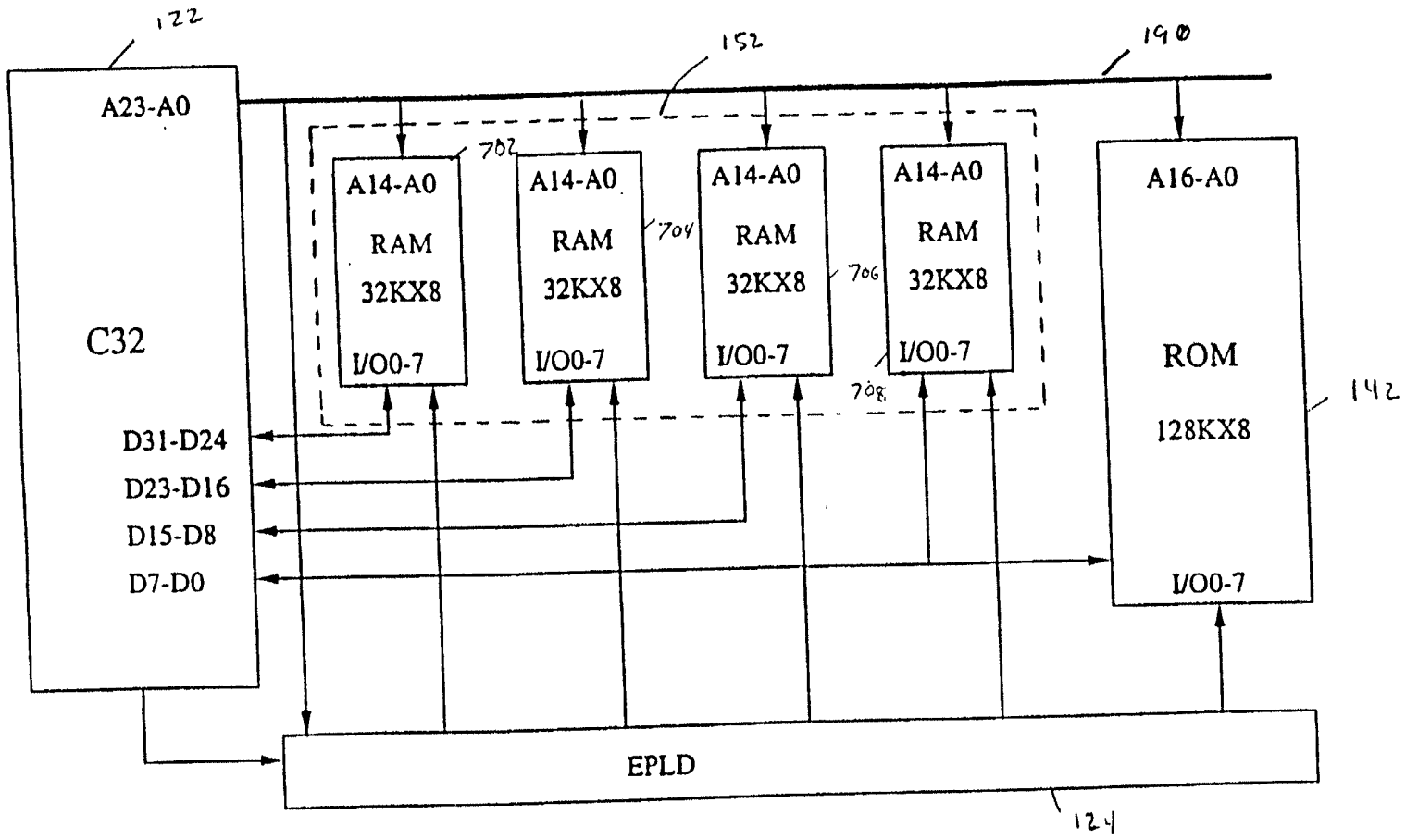


FIG 7

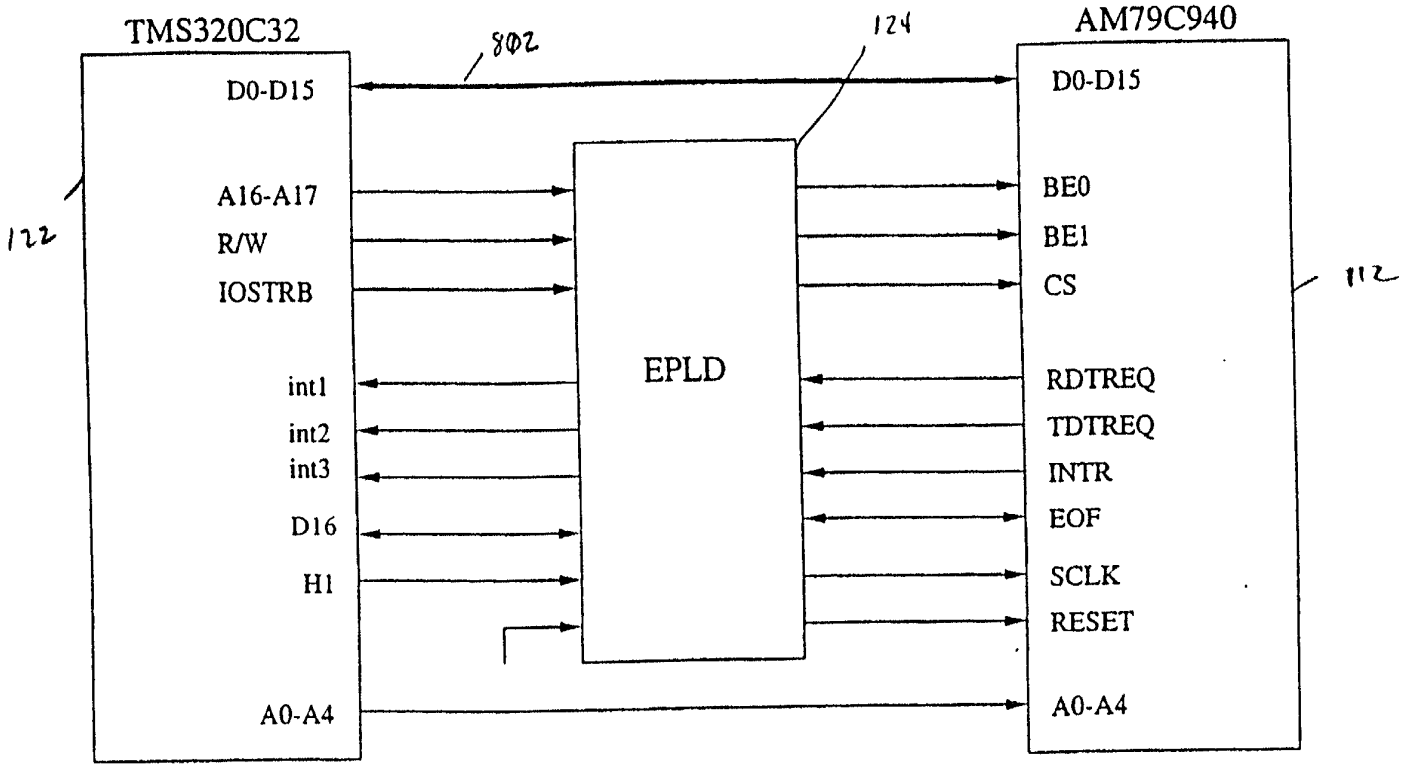


FIG. 8

900

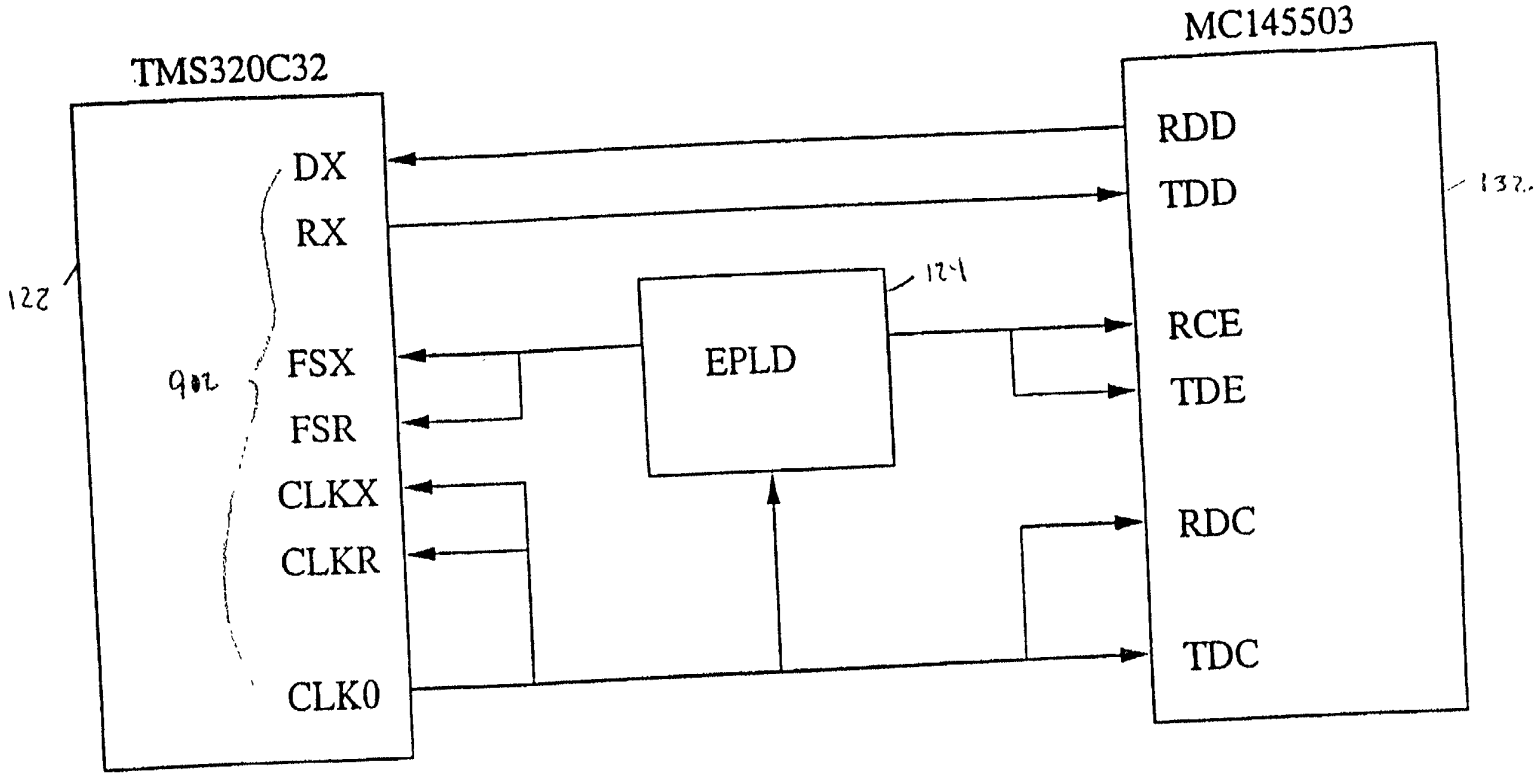
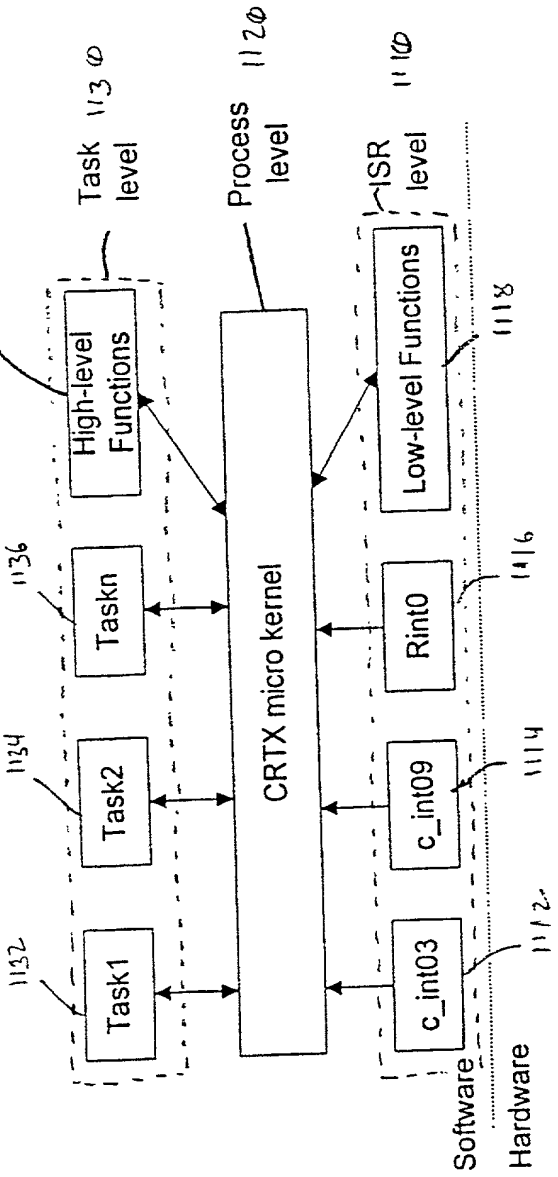


FIG. 9

DSP Address	Usage
0x2000	Command port for left-half of LCD
0x20001	Data port for left-half of LCD
0x2002	Command port for right-half of LCD
0x2003	Data port for right-half of LCD

FIG. 10


1100
Task level
1130
Process level
1120
ISR level
1110
1138
High-level Functions
1136
Taskn
1134
Task2
1132
Task1
CRTX micro kernel
Low-level Functions
1118
Rint0
1116
c_int09
1114
c_int03
1112
Software
Hardware

1200
1120
CRTX
1222
Real-time Event Flag Manager
1224
Delayed Task Manager
1226
Scheduling Manager

FIG. 12

FIG. 13A

Task Level Software

Function Name	Function
ARPinit()	(Initialization Function) ARP table initialization.
c_int00()	(Initialization Function) Main program, initialize the stack pointer, external bus interface, and interrupt vector for TMS320C32.
DMA_initialize()	(Initialization Function) Initialize the DMA0 and DMA1 channels.
ENET_initialize()	(Initialization Function) Initialize the Ethernet controller.
InitHardWare()	(Initialization Function) Initialize the Timer0, Timer1 and serial port.
NameInit()	(Initialization Function) Initialize some SIP headers and SDP body.
SerialPortInit()	(Initialization Function) Initialize the serial port.
ARP_In_task()	Parse ARP input packets
ARPTimer_task()	ARP timer, maintain the ARP table
Call_task()	Call processing
Clock_task()	A clock generates the hour, minute, and second
Codec_task()	A task to call encoding, decoding, ring generation, tone generation or memory loop.
CreateSipCall()	Create a SIP request packet for a call
Ercv_task()	Ethernet packet receiver and IP de-multiplexing.

FIG. 13B

Function Name	Function
IP_Send_task()	IP multiplexing and Ethernet packet sending.
Key_task()	Key pad monitor and input.
RTP_In_task()	RTP processing.
Sendto()	Send UDP packets to given IP address.
Setting_task()	Setting the E*Phone parameters.
SIP_In_task()	Accept SIP packets, and update call and SIP status.
SIP_task()	SIP status transition task
Tone_task()	Count the active and stop duration for tone or ring.
UDP_In_task()	Accept UDP packets
ARP_Out()	(High-level function) ARP request program
ClearScreen()	(High-level function) Clear all lines on the LCD
CodecConfig()	(High-level function) Schedule a codec task according to the run mode parameter
Disp()	(High-level function) Display a string on the LCD screen
LCD()	(High-level function) Display a character on the LCD screen
LCDClear()	(High-level function) Clear one line on the LCD screen
LinearToUlaw()	(High-level function) Linear data to u-law data conversion

FIG. 13C

Function Name	Function
Initialization()	(High-level function) Call initialization function and pre-schedule tasks
RTP_para_init()	(High-level function) Generate the random time stamp and SSRC for a RTP session
ScreenScroll()	(High-level function) Scroll the LCD screen for one line upward or downward
SDPParse()	(High-level function) Parse SDP packets
SIPParse()	(High-level function) Parse SIP packets
SIP_Request()	(High-level function) Create SIP request messages
SIP_Response()	(High-level function) Create SIP response messages
SpeechDecode()	(High-level function) Speech decoding
SpeechEncode()	(High-level function) Speech encoding
ToneGenerate()	(High-level function) Generates dial tone, ring back tone, busy tone or alert tone.

FIG. 13D

Process Level Software

Function
Function prototypes
Header file
Supervisor (kernel)
Trap Manager source file

FIG. 13E

ISR Level Software

Function Name	Function
c_int03()	Ethernet controller ISR. Triggered on INT3 of TMS320C32 by external interrupt from AM79C940.
c_int09()	System timer ISR. Triggered on TINT1 by internal timer1 of TMS320C32.
Rint0()	A/D and D/A ISR. Triggered on RINT0 by internal serial port interrupt of TMS320C32.
AmpControl()	(Low-level function) Control the speaker volume.
DMA1()	(Low-level function) Start the DMA1 channel.
DMA0_Release()	(Low-level function) Start the DMA0 channel.
DMA_int_set()	(Low-level function) Enable INT1 and INT2 for DMA0 and DMA1.
ENET_reset()	(Low-level function) Reset the Ethernet controller.
ENET_disable()	(Low-level function) Disable the Ethernet controller.
HandSet()	(Low-level function) Control the handset and hands-free switching.
HookState()	(Low-level function) Check the hook state.
Key()	(Low-level function) Key pad check and read.
KeyMap()	(Low-level function) Map the key binary input to ASCII format.
LCDCmd()	(Low-level function) LCD control command.

FIG 13F

Function Name	Function
LCDWrite()	(Low-level function) Write display data to LCD.
RintEnable()	(Low-level function) Enable the RINT0 for Rint0 ISR.
RintDisable()	(Low-level function) Disable the RINT0.
SerialPortRst()	(Low-level function) Reset the serial port.
TimerEnable()	(Low-level function) Enable the system timer TCLK1.
TimerDisable()	(Low-level function) Disable the system timer TCLK1.

1400

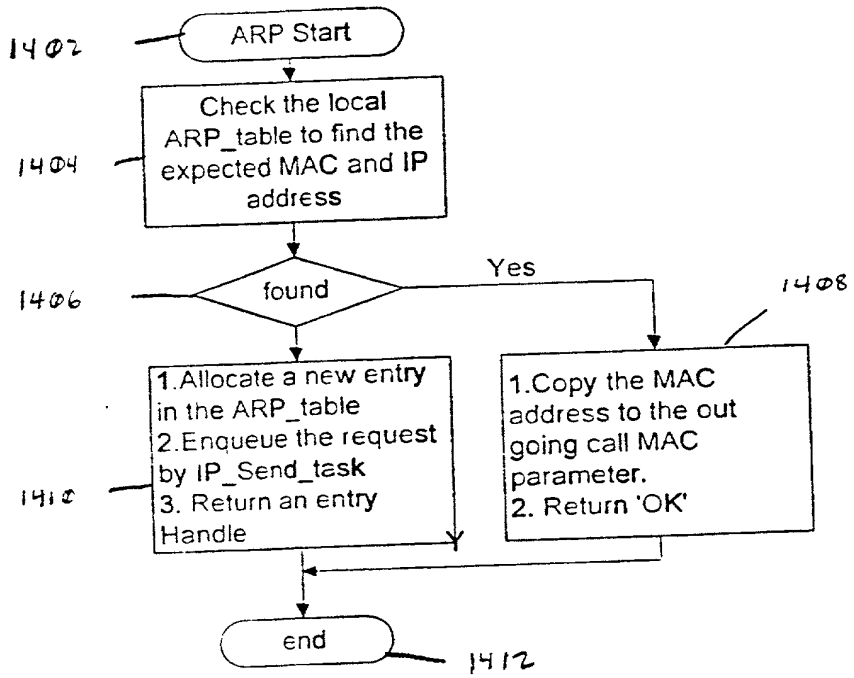

1402
ARP Start
Check the local ARP_table to find the expected MAC and IP address
1404
found
1406
Yes
1408
1.Allocate a new entry in the ARP_table
2.Enqueue the request by IP_Send_task
3. Return an entry Handle
1410
1.Copy the MAC address to the out going call MAC parameter.
2. Return 'OK'
end
1412

FIG. 14

1500

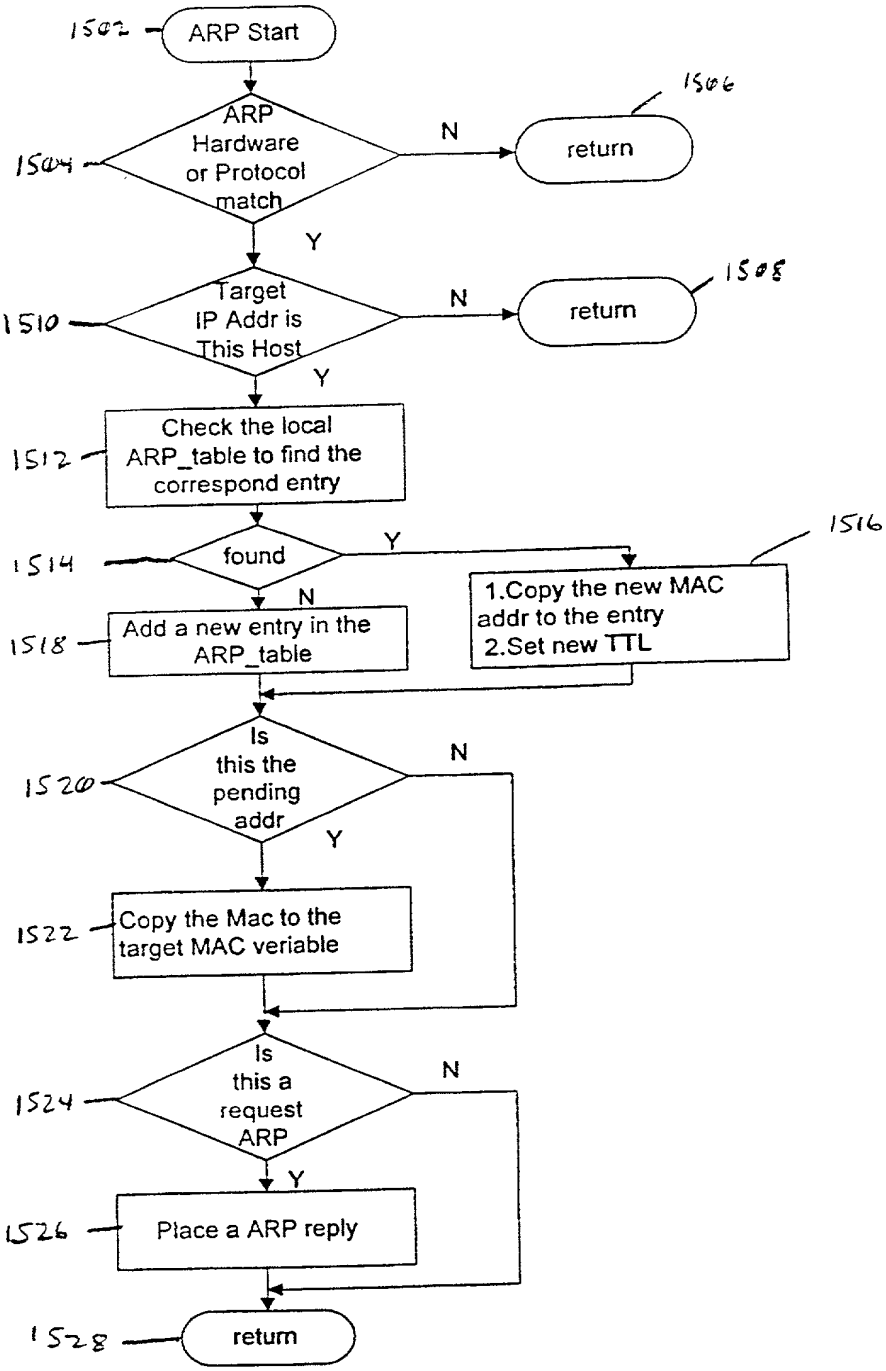


FIG.15

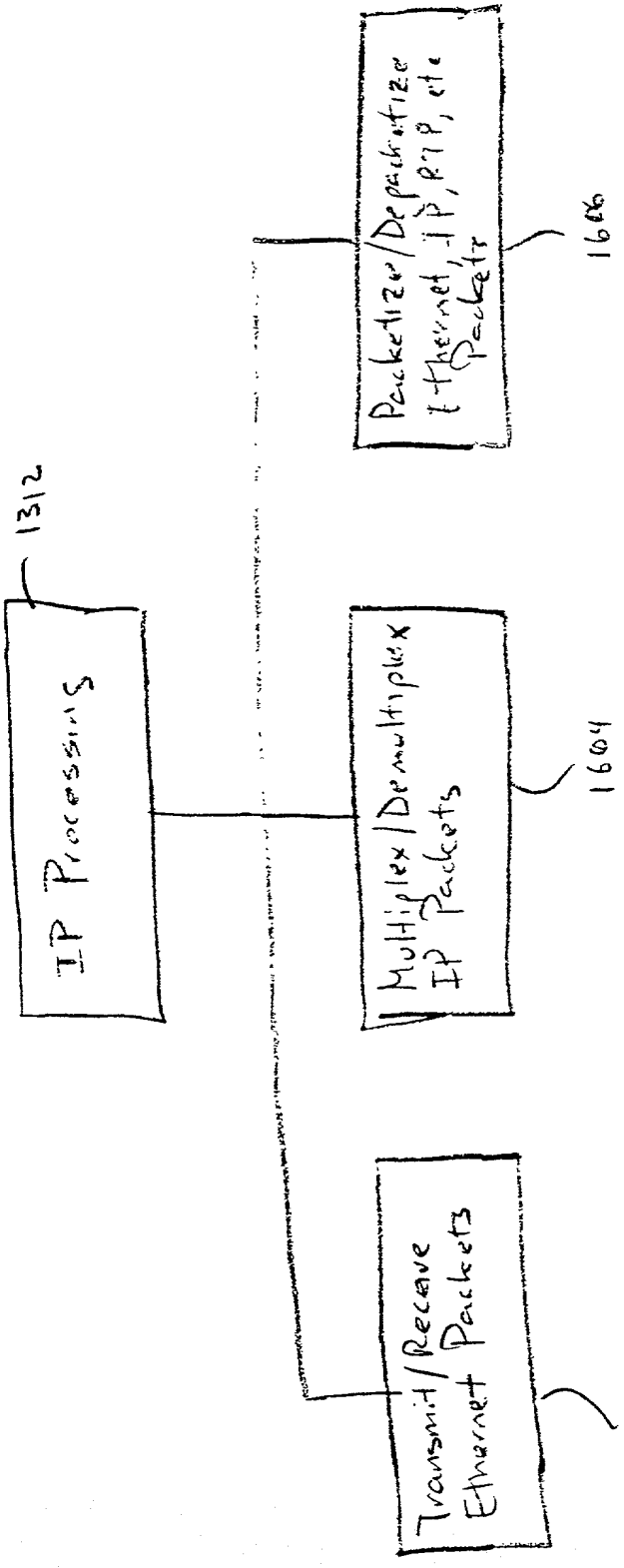

IP Processors
1312
Multiplex/Demultiplex IP Packets
1604
Transmit/Receive Ethernet Packets
1602
Packetizer/Depacketizer Ethernet, IP, RTP, etc Packets
1606

FIG. 16

```
struct ENetHeader {          /* Ethernet header structure */
  ETA Dest;                  /* Ethernet destination MAC address */
  ETA Source;                /* Source MAC address */
  int Type;                  /* Ethernet packet type */
};

struct IPHeader {            /* IP header structure */
  int VI_ToS;                /* IP version, header length, and service type */
  int Length;                /* total length */
  int Identify;              /* identifier of the IP packet */
  int FragDff;               /* flags and fragment offset */
  int TTL_Protocol;          /* time-to-live, and protocols */
  int ChkSum;                     /* checksum */
  IPA Source;                /* source IP address */
  IPA Dest;                  /* destination IP address */
};

struct UDPHeader {           /* UDP header structure */
  int SPort;                 /* source port */
  int DPort;                 /* destination port */
  int Length;                /* UDP message length */
  int ChkSum;                     /* UDP checksum */
};

struct EPACKET {            /* Ethernet receive packet structure */
  struct ENetHeader Enh;    /* Ethernet header */
  struct IPHeader Iph;            /* IP header */
  struct UDPHeader Uh;      /* UDP header */
  int data[MaxUDPLength];   /* data field */
};
```

FIG. 17

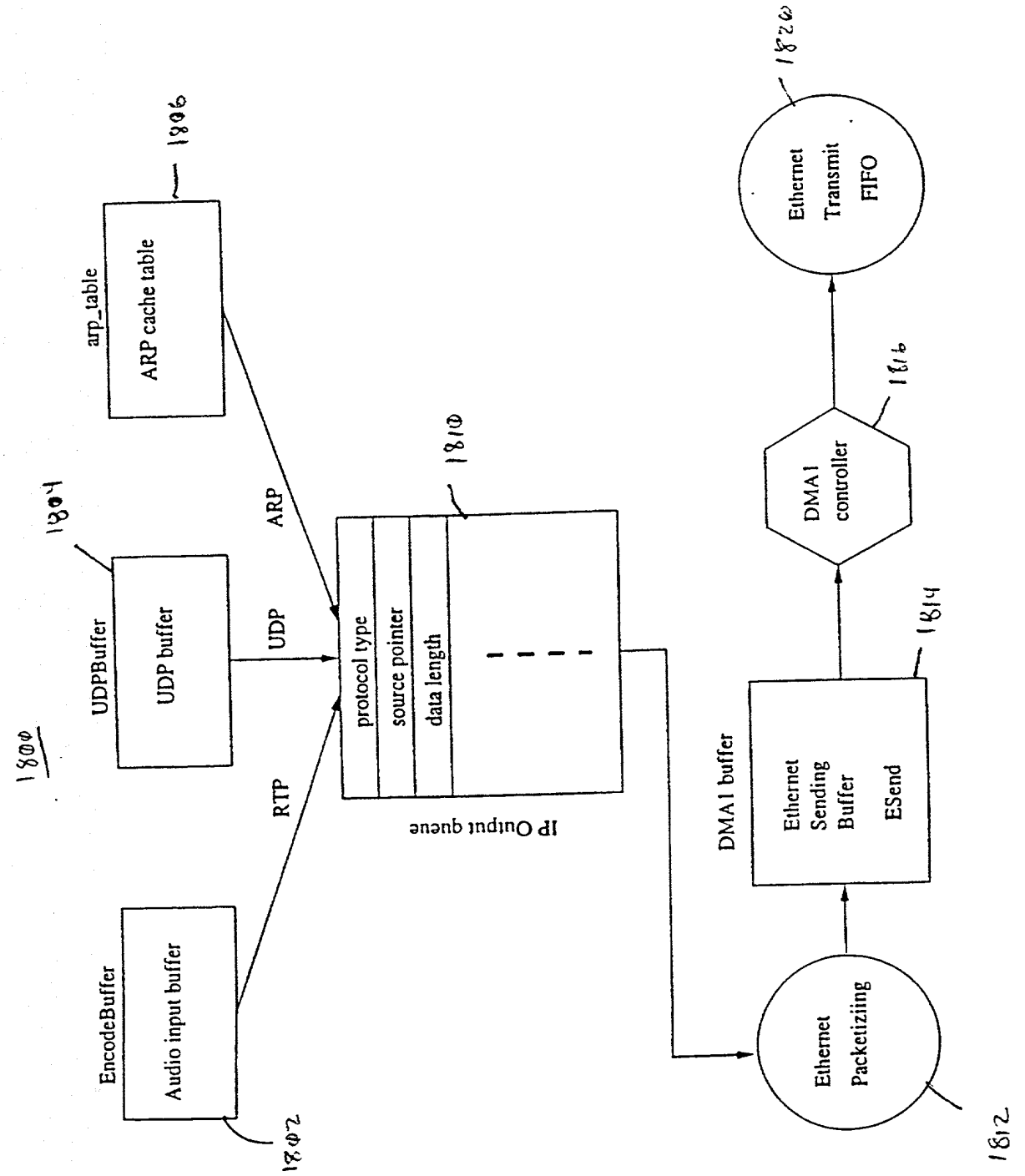

1800
1802
EncodeBuffer
Audio input buffer
RTP
1804
UDPBuffer
UDP buffer
UDP
1806
arp_table
ARP cache table
ARP
1810
protocol type
source pointer
data length
IP Output queue
1812
Ethernet Packetiziing
DMA1 buffer
1814
Ethernet Sending Buffer
ESend
1816
DMA1 controller
1820
Ethernet Transmit FIFO

FIG. 18

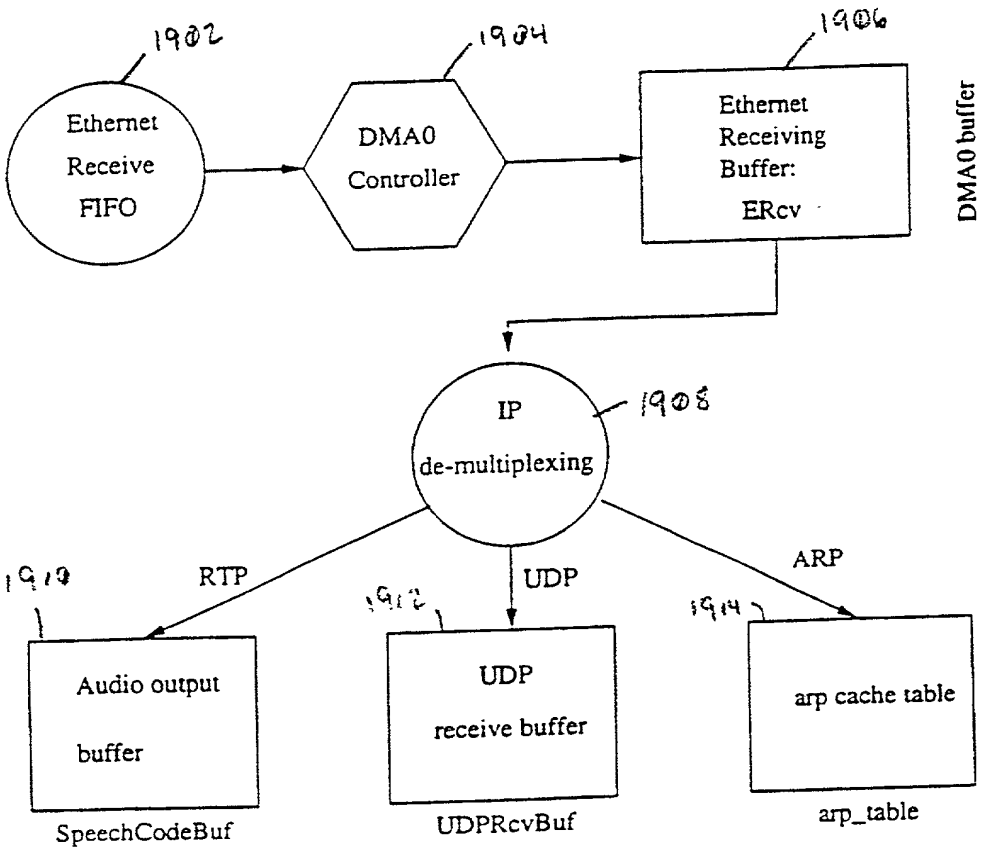

1900
1902
Ethernet Receive FIFO
1904
DMA0 Controller
1906
Ethernet Receiving Buffer: ERcv
DMA0 buffer
IP de-multiplexing
1908
RTP
UDP
ARP
1910
Audio output buffer
SpeechCodeBuf
1912
UDP receive buffer
UDPRcvBuf
1914
arp cache table
arp_table

FIG. 19

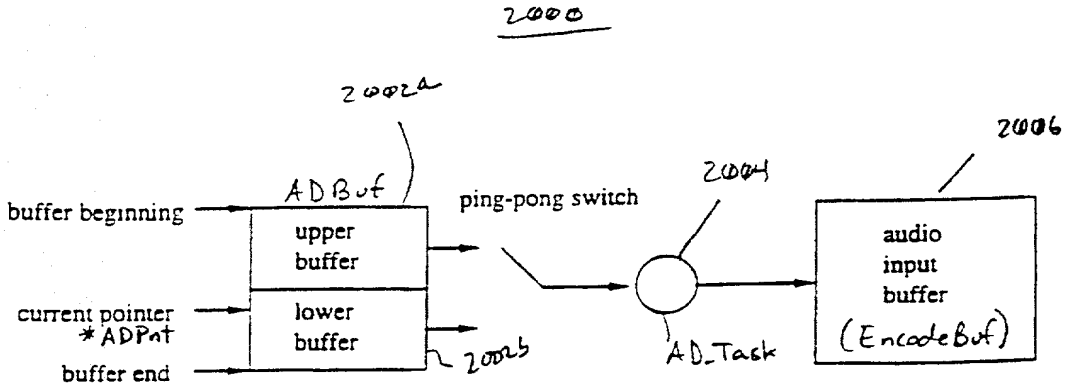


FIG. 20A

(a) A/D buffer update scheme

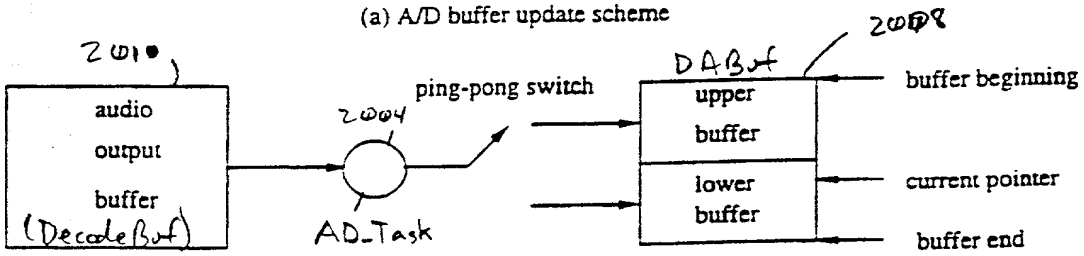


FIG. 20B

2100

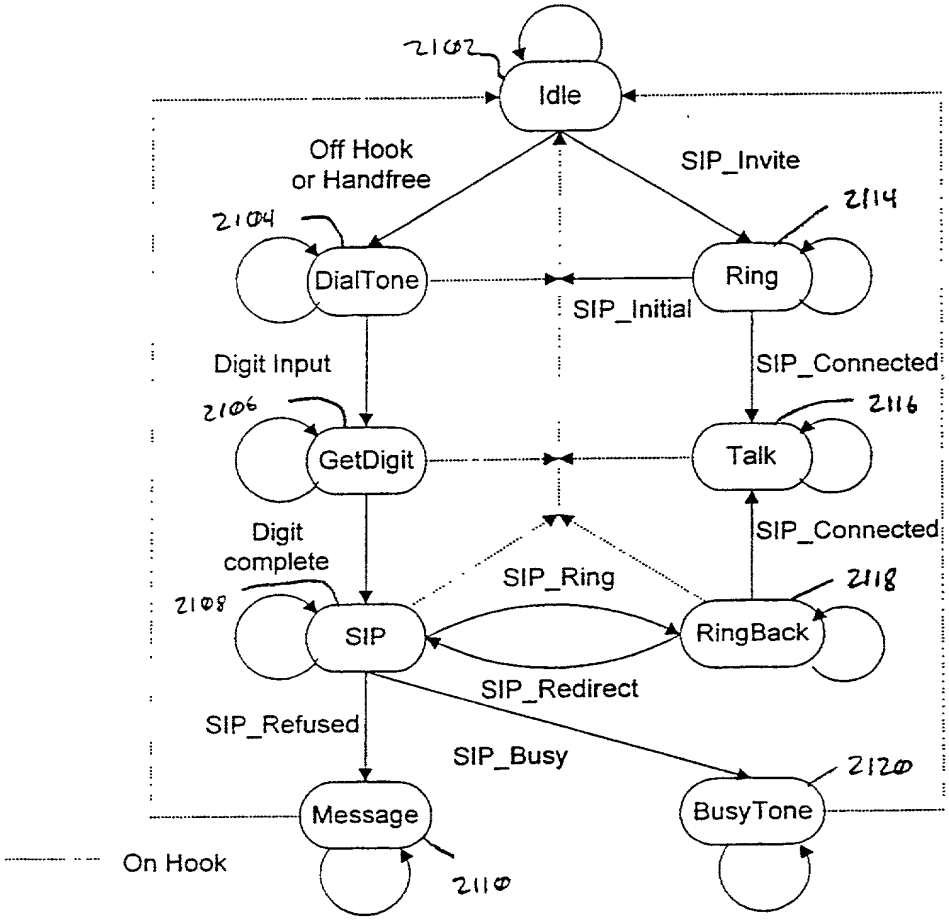

2102
Idle
Off Hook
or Handfree
SIP_Invite
2104
2114
DialTone
Ring
SIP_Initial
Digit Input
SIP_Connected
2106
2116
GetDigit
Talk
Digit
complete
SIP_Connected
SIP_Ring
2118
2108
SIP
RingBack
SIP_Redirect
SIP_Refused
SIP_Busy
2120
Message
BusyTone
On Hook
2110

FIG. 21

FIG. 22

Keys	ReturnValue
Digit keys	'0' ... '9'
Special keys	'*' and '#'
Enter key	'E'
Hands Free	'H'
Redial key	'R'
Upward	'U'
Downward	'D'

FIG. 23

```
struct FuncKey {
  WORD Enter;      Enter key
  WORD Redial;     Redial key
  WORD Up;         Up arrow key
  WORD Down;       Down arrow key
  WORD Digit;      digit keys or special key
  WORD Full;             key buffer full
  WORD Enable;     when set, indicates key input is enabled
  WORD Touch;      any key was pressed
  WORD Alf;        an alphabetic key was pressed.
};
```

FIG 24

Port address	Read Write	Bit Value	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x820001	W	0	x[5]	x	x	x	Soft reset	Volume lock	ENET release	Hand free
0x820001	W	1	x	x	x	x	x	unlock	ENET reset	Hand set
0x820001	R	0	x	x	x	x	x	No key input	x	Hook off
0x820001	R	1	x	x	x	x	x	Key touched	x	Hook on

FIG. 25

```
struct Message {        /* a structure for all messages in the SIP Phone */
  int ENetXmtST;        /* Ethernet transmission  packet state */
  int ENetRcvST0;       /* Ethernet receiving packet state */
  int ENetRcvST1;       /* Ethernet receiving packet state */
  int ENetRcvST2;       /* Ethernet receiving packet state */
  int ENetRcvST3;       /* Ethernet receiving packet state */
  int RcvFlag;          /* The receiving speech data is available when SET */
  int ARPST;            /* reserved */
};
```

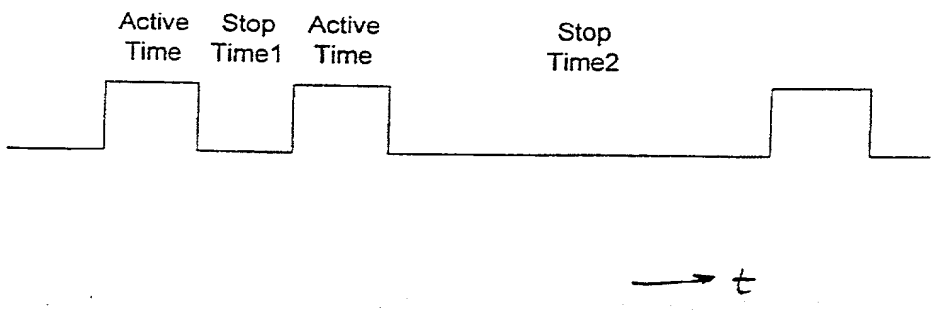
FIG. 26

```
typedef struct {
  int pt:7        /* payload type */
  int m:1         /* marker bit */
  int cc:4        /* CSRC count */
  int x:1         /* header extension flag */
  int p:1         /* padding flag */
  int version:2 /* protocol version */
  int seq         /* sequence number */
  int ts1         /* timestamp least significant 16 bits */
  int ts2         /* timestamp most significant 16 bits */
  int ssrc1       /* Synchronization source least significant 16 bits */
  int ssrc2       /* Synchronization source most significant 16 bits */
  int csrc[1]     /* optional CSRC list address */
} RTPHeader;
```

FIG. 27

```
struct ToneType {
        int ActiveTime;      The period for sound is active
        int ActiveCnt;       The counter for the sound during the active time
        int StopTime1;       First sound stop period
        int StopCnt1;        First sound stop counter
        int StopTime2;       Second sound stop period
        int StopCnt2;        Second sound stop counter
}
```

FIG. 28



```
typedef struct {          /* string type used in message_t structure */
  char *s;                /* start of string */
  short len;              /* length of string */
} string;

typedef enum {            /* state transition structure */
   Initial,               /* SIP initial state, UAC or UAS */
   Proceeding,            /* proceeding of the request, UAS */
   Failure,               /* failure, UAS */
   Success,               /* success, UAS */
   Confirmed,             /* confirmed, UAS */
   Calling,               /* calling, UAC */
   CallProc,              /* call proceeding, UAC */
   Completed,             /* completed, UAC */
   Bye                    /* Bye state, UAC or UAS */
} Tstate;

typedef struct {          /* SIP message structure
  method_t method;        /* request: method; response: 0 */
  short status;           /* response: status value; request: 0 */
  string url;             /* request URL */
  string via;             /* via header */
  string callid;          /* Call-ID */
  string contact;         /* contact header */
  string from;            /* From address */
  string from_display;    /* From display name */
  string subject;         /* Subject */
  string to;              /* To address */
  string to_display;      /* To display name */
  string ts;              /* timestamp */
  string reason;          /* response reason phrase */
  content_t contenttype;/* contact type header */
  int contentlength;      /* contact length */
  unsigned cseq;          /* sequence number */
  string body;            /* SDP body */
  sdp_t sdp;              /* session description */
} message_t;

typedef struct {          /* call structure */
  int flag;               /* SET for effective, RESET for clear */
  int ua_state;           /* Not current call:0; UAC:1; UAS:2 */
  int status;             /* current response status */
  message_t m;            /* SIP message */
  char * udp;             /* receive SIP packets pointer */
  char * local;           /* UAC request packet pointer */
  sockaddr peer;          /* peer host IP address */
  sdp_t sdp;              /* sdp backup */
  Tstate state;           /* SIP transition state */
  int t1;                 /* T1 timer */
  int t2;                 /* T2 timer */
} call;
```

FIG. 29

FIG. 30

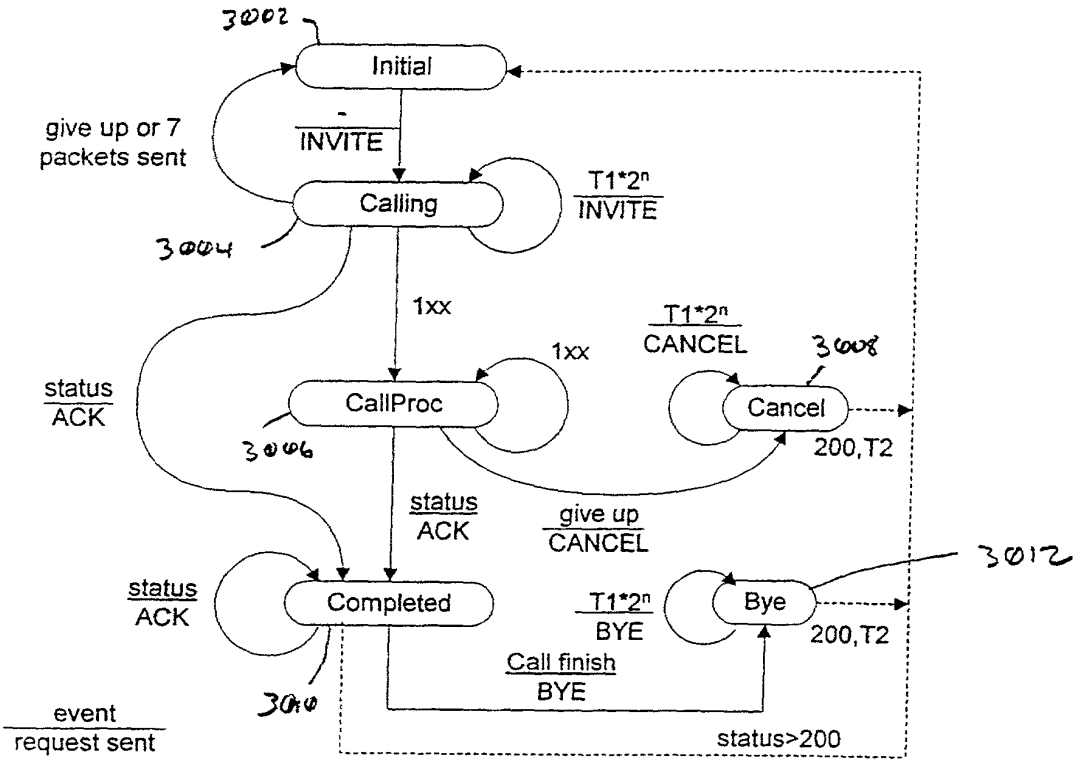


FIG. 31

Message received	SIP_Status
100	SIP_Trying
18x	SIP_Ring
200	SIP_Connected
3xx	SIP_Redirect
4xx, 5xx	SIP_Refused
6xx	SIP_Busy

Fig. 32

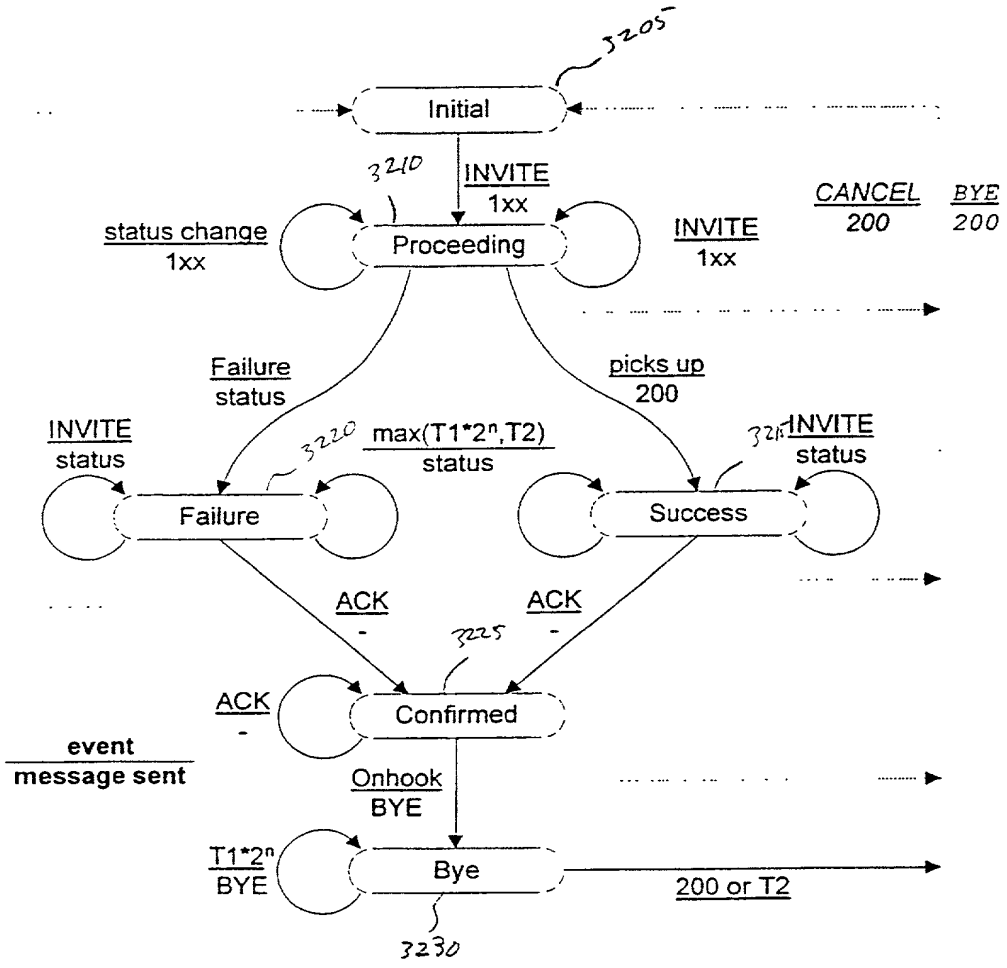
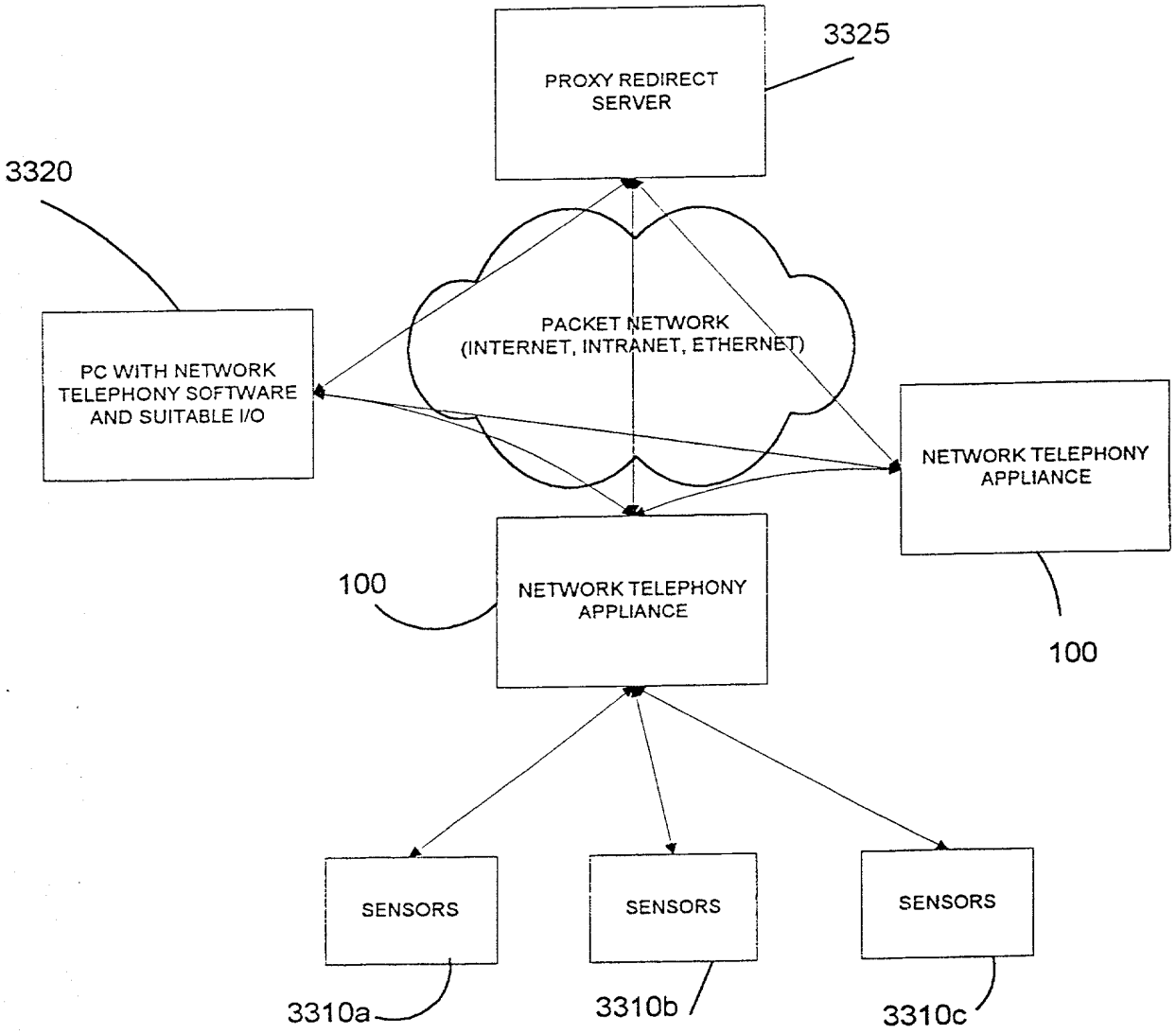

3205
Initial
3210
INVITE
1xx
status change
1xx
Proceeding
INVITE
1xx
CANCEL
200
BYE
200
Failure
status
picks up
200
INVITE
status
3220
max(T1*2^n,T2)
status
3215
INVITE
status
Failure
Success
ACK
-
ACK
-
3225
ACK
-
Confirmed
event
message sent
Onhook
BYE
T1*2^n
BYE
Bye
200 or T2
3230

FIG. 33


PROXY REDIRECT SERVER
3325
3320
PACKET NETWORK (INTERNET, INTRANET, ETHERNET)
PC WITH NETWORK TELEPHONY SOFTWARE AND SUITABLE I/O
NETWORK TELEPHONY APPLIANCE
100
NETWORK TELEPHONY APPLIANCE
100
SENSORS
SENSORS
SENSORS
3310a
3310b
3310c